Weighted Model Counting with Conditional Weights

Paulius Dilkas

30th July 2020

1 Introduction

- The Main Narrative
  1. When weights are defined on literals, the measure on the free BA is fully independent.
  2. This means that the BA itself must be larger (i.e., have additional 'meaningless' literals) to turn any probability distribution into an independent one.
  3. We show how we can define conditional weights on literals, allowing us to encode any probability distribution into a Boolean algebra that's not necessarily independent and thus can be smaller.
  4. We demonstrate a specific example of this by presenting a new way to encode Bayesian networks into instances of WMC and adapting a WMC algorithm (ADDMC) to run on the new format.
  5. We show that this results in significantly faster inference.
  6. We show that our encoding results in asymptotically fewer literals and fewer ADDs.
- Potential criticism may be that this doesn't allow us to use SAT-based techniques for probabilistic inference. However, they can still be used for a significant part of the encoding.
  - Zero-probability weights and one-probability weights can be interpreted as logical clauses. This doesn't affect ADDMC but could be useful for other solvers.

F What are the main claims, what are the main takeaways, intuitive [???] of theorems to follow.

- Algorithms
  - ADDMC [12] (rediscovered the multiplicativity of BAs in different words) (with optimal settings)
  - Cachet [27]
  - c2d [10]
  - d4 [21] (closed source, boo!)
  - miniC2D [26]
- Notable previous/related work
  - Hailperin's approach to probability logic [15]
  - Nilsson's (somewhat successful) probabilistic logic [24, 25]
  - Measures on Boolean algebras
    - On possibility and probability measures in finite Boolean algebras [2]
    - Representation of conditional probability measures [20]
- Intuitively, a measure is just like a probability, except it's in $\mathbb{R}_{\geq 0}$ instead of $[0, 1]$.

Table 1: A comparison of Boolean-algebraic (BA) and set-theoretic (ST) concepts for 2^X for some set X

BA name	BA symbol	ST symbol	ST name
bottom	$\bot$	$\emptyset$	empty set
top	$\top$	X	
meet, and	$\wedge$	$\cap$	intersection
join, or	$\vee$	$\cup$	union
complement, not	$\neg$	^{c}	complement
	$\leq$	$\subseteq$	subset relation, set inclusion
atom			singleton, unit set

2 Boolean Algebras and Power Sets

Let X be a set and let 2^X be its power set. We can equivalently interpret 2^X as a Boolean algebra. See Table 1 for a summary of the differences in terminology and notation. We will use both.

2.1 The Space of Functions on Boolean Algebras

Definition 1 (Operations on functions)**.** Let $A\colon 2^X \to \mathbb{R}_{\geq 0}$ and $B\colon 2^Y \to \mathbb{R}_{\geq 0}$ be functions, $\alpha \in \mathbb{R}_{\geq 0}$, and $x \in X$. We define the following operations:

Addition: $A+B\colon 2^{X\cup Y} \to \mathbb{R}_{\geq 0}$ is a function such that $(A+B)(\tau) = A(\tau\cap X)+B(\tau\cap Y)$ for all $\tau \in 2^{X\cup Y}$.

Inverse: $\overline{A}\colon 2^X \to \mathbb{R}_{\geq 0}$ is a function such that $\overline{A}(\tau) = 1 - A(\tau)$ for all $\tau \in 2^X$.

Multiplication: $A \cdot B\colon 2^{X\cup Y} \to \mathbb{R}_{\geq 0}$ is a function such that $(A \cdot B)(\tau) = A(\tau \cap X) \cdot B(\tau \cap Y)$ for all $\tau \in 2^{X\cup Y}$.

Scalar multiplication: $\alpha A\colon 2^X \to \mathbb{R}_{\geq 0}$ is a function such that $(\alpha A)(\tau) = \alpha \cdot A(\tau)$ for all $\tau \in 2^X$.

Projection: $\exists_x A\colon 2^{X\setminus\{x\}} \to \mathbb{R}_{\geq 0}$ is a function such that $(\exists_x A)(\tau) = A(\tau) + A(\tau \cup \{x\})$ for all $\tau \in 2^{X\setminus\{x\}}$.

Observation 1. *Let U be a set, and $\mathcal{V} = \{A\colon 2^X \to \mathbb{R}_{\geq 0} \mid X \subseteq U\}$. Then $\mathcal{V}$ is a semi-vector space with three additional operations: inverse, (non-scalar) multiplication, and projection. Specifically, note that both addition and multiplication are both associative and commutative.*

Definition 2 (Special functions)**.**

- unit $1\colon 2^\emptyset \to \mathbb{R}_{\geq 0}$, $1(\emptyset) = 1$.
- zero $0\colon 2^\emptyset \to \mathbb{R}_{\geq 0}$, $0(\emptyset) = 0$.
- logical constant $[a]\colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$, $[a](\emptyset) = 0$, $[a](\{a\}) = 1$.

Henceforth, for any function $A\colon 2^X \to \mathbb{R}_{\geq 0}$ and any set τ, we will write $A(\tau)$ to mean $A(\tau \cap X)$.

3 Weighted Model Counting as a Measure

Definition 3 (Measures and weight functions)**.** Let U be a set.

- A *measure* is a function $M\colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ such that $M(\bot) = 0$, and $M(a \vee b) = M(a) + M(b)$ for all $a, b \in 2^{2^U}$ whenever $a \wedge b = \bot$.
- A *weight function* is a function $W\colon 2^U \to \mathbb{R}_{\geq 0}$.

Table 2: An overview of notation for a logic defined over two atoms. The elements in both columns are listed in the same order.

Name in logic	Boolean-algebraic notation	Set-theoretic notation
Atoms (elements of U)	a, b	a, b
Models (elements of 2^U)	$\neg a \wedge \neg b, a \wedge \neg b, \neg a \wedge b, a \wedge b$	$\emptyset, \{a\}, \{b\}, \{a, b\}$
Formulas (elements of 2^{2^U})	$\top$	$\{\emptyset, \{a\}, \{b\}, \{a, b\}\}$
	$\neg a \vee \neg b, a \to b$	$\{\emptyset, \{a\}, \{b\}\}, \{\emptyset, \{a\}, \{a, b\}\}$
	$b \to a, a \vee b$	$\{\emptyset, \{b\}, \{a, b\}\}, \{\{a\}, \{b\}, \{a, b\}\}$
	$\neg b, \neg a, a \leftrightarrow b$	$\{\emptyset, \{a\}\}, \{\emptyset, \{b\}\}, \{\emptyset, \{a, b\}\}$
	$(a \wedge \neg b) \vee (b \wedge \neg a), a, b$	$\{\{a\}, \{b\}\}, \{\{a\}, \{a, b\}\}, \{\{b\}, \{a, b\}\}$
	$\neg a \wedge \neg b, a \wedge \neg b, \neg a \wedge b, a \wedge b$	$\{\emptyset\}, \{\{a\}\}, \{\{b\}\}, \{\{a, b\}\}$
	$\bot$	$\emptyset$

- A weight function is *factored* if $W = \prod_{x \in U} W_x$ for some functions $W_x \colon 2^{\{x\}} \to \mathbb{R}_{\geq 0}$, $x \in U$.
- We say that a weight function $W \colon 2^U \to \mathbb{R}_{\geq 0}$ *induces* a measure $M_W \colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ if

$$M_W(x) = \sum_{\{u\} \leq x} W(u). \tag{1}$$

- A measure $M \colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ is *factorable* if there exists a factored weight function $W \colon 2^U \to \mathbb{R}_{\geq 0}$ that induces M.

Lemma 1. *The function M_W, as defined by Eq.* (1)*, is a measure.*

Proof. Note that $M_W(\bot) = 0$ since there are no atoms below $\bot$. Let $a, b \in 2^{2^U}$ be such that $a \wedge b = \bot$. By elementary properties of Boolean algebras, all atoms below $a \vee b$ are either below a or below b. Moreover, none of them can be below both a and b because then they would have to be below $a \wedge b = \bot$. Thus

$$M_W(a \vee b) = \sum_{\{u\} \leq a \vee b} W(u) = \sum_{\{u\} \leq a} W(u) + \sum_{\{u\} \leq b} = M_W(a) + M_W(b)$$

as required. □

In this formulation, the process of calculating the value of $M_W(x)$ for some $x \in 2^{2^U}$ with a given definition of W is known as *weighted model counting*.

3.1 Relation to the Classical (Logic-Based) View of WMC

Would it be more accurate to replace the word 'model' with 'interpretation' in most situations?

Let $\mathcal{L}$ be a propositional logic with atoms $U = \{a, b\}$. See Table 2 for an overview of the models and formulas in $\mathcal{L}$ as well as their set-theoretic and Boolean-algebraic representations. Note the differences in the meaning of the word 'atom' between the logical and the Boolean-theoretic interpretations. In the Boolean algebra 2^{2^U}, an atom is a set with a single element which corresponds to a model of $\mathcal{L}$ (also known as an element of 2^U); whereas an atom of $\mathcal{L}$ is an atomic formula, i.e., an element of U. We will primarily use the term 'atom' to mean the former.

Let $w \colon \{a, b, \neg a, \neg b\} \to \mathbb{R}_{\geq 0}$ be the *weight function* defined by

$$w(a) = 0.3, \quad w(\neg a) = 0.7, \quad w(b) = 0.2, \quad w(\neg b) = 0.8.$$

Let Δ be a theory in $\mathcal{L}$ with a sole axiom a. Then Δ has two models, i.e., $\{a, b\}$ and $\{a, \neg b\}$. The *weighted model count* (WMC) [6] of Δ is then

$$\mathrm{WMC}(\Delta) = \sum_{\omega \models \Delta} \prod_{\omega \models l} w(l) = w(a)w(b) + w(a)w(\neg b) = 0.3. \tag{2}$$

Alternatively, we can define $W_a \colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ as

$$W_a(\{a\}) = 0.3, \quad W_a(\emptyset) = 0.7$$

and $W_b \colon 2^{\{q\}} \to \mathbb{R}_{\geq 0}$ as

$$W_b(\{b\}) = 0.2, \quad W_b(\emptyset) = 0.8.$$

Let M be the measure on 2^{2^U} induced by $W = W_a \cdot W_b$. Then, equivalently to Eq. (2), we can write

$$M(a) = W(\{a, b\}) + W(\{a\}) = W_a(\{a\})W_b(\{b\}) + W_a(\{a\})W_b(\emptyset) = 0.3.$$

Given the theory Δ, we can also compute the probability of a query b as [28]

$$\Pr_{\Delta, w}(b) = \frac{\mathrm{WMC}(\Delta \wedge b)}{\mathrm{WMC}(\Delta)}.$$

The same thing can be accomplished using the algebraic formulation, with M replacing WMC.

In the rest of the paper, for any set U, we will use set-theoretic notation for 2^U and Boolean-algebraic notation for 2^{2^U}, except for (Boolean) atoms in 2^{2^U} that are denoted as $\{x\}$ for some model $x \in 2^U$.

3.2 Limitations of Factorable Measures

Clarify what is meant by'conditional weight function'.

Example 1. Let $U = \{a, b\}$ be a set of atoms and $M \colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ a measure defined as:[1]

$$\begin{aligned} M(a \wedge b) &= 0.72, \\ M(a \wedge \neg b) &= 0.18, \\ M(\neg a \wedge b) &= 0.07, \\ M(\neg a \wedge \neg b) &= 0.03. \end{aligned}$$

If M could be represented using traditional (factored) WMC, we would have to find two weight functions $W_a \colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ and $W_b \colon 2^{\{b\}} \to \mathbb{R}_{\geq 0}$ such that $W = W_a \cdot W_b$ induces M, i.e., W_a and W_b would have to satisfy this system of equations:

$$\begin{aligned} W_a(\{a\}) \cdot W_b(\{b\}) &= 0.72 \\ W_a(\{a\}) \cdot W_b(\emptyset) &= 0.18 \\ W_a(\emptyset) \cdot W_b(\{b\}) &= 0.07 \\ W_a(\emptyset) \cdot W_b(\emptyset) &= 0.03, \end{aligned}$$

which has no solutions.

Alternatively, we can let b depend on a and consider weight functions $W_a \colon 2^{\{a\}} \to \mathbb{R}_{\geq 0}$ and $W_b \colon 2^{\{a,b\}} \to \mathbb{R}_{\geq 0}$ defined as

$$\begin{aligned} W_a(\{a\}) &= 0.9, \\ W_a(\emptyset) &= 0.1, \end{aligned}$$

[1] The value of M on any other element of the Boolean algebra can be deduced using the definition.

and

$$\begin{aligned} W_b(\{a,b\}) &= 0.8, \\ W_b(\{a\}) &= 0.2, \\ W_b(\{b\}) &= 0.7, \\ W_b(\emptyset) &= 0.3. \end{aligned}$$

One can easily check that with these definitions W indeed induces M.

Lemma 2. *For any measure $M\colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ and elements $a, b \in 2^{2^U}$,*

$$M(a \wedge b) = M(a)M(b) \tag{3}$$

if and only if

$$M(a \wedge b) \cdot M(\neg a \wedge \neg b) = M(a \wedge \neg b) \cdot M(\neg a \wedge b). \tag{4}$$

Proof. First, note that $a = (a \wedge b) \vee (a \wedge \neg b)$ and $(a \wedge b) \wedge (a \wedge \neg b) = 0$, so, by properties of a measure,

$$M(a) = M(a \wedge b) + M(a \wedge \neg b). \tag{5}$$

Applying Eq. (5) and the equivalent expression for $M(b)$ allows us to rewrite Eq. (3) as

$$M(a \wedge b) = [M(a \wedge b) + M(a \wedge \neg b)][M(a \wedge b) + M(\neg a \wedge b)]$$

which is equivalent to

$$M(a \wedge b)[1 - M(a \wedge b) - M(a \wedge \neg b) - M(\neg a \wedge b)] = M(a \wedge \neg b)M(\neg a \wedge b). \tag{6}$$

Since $a \wedge b$, $a \wedge \neg b$, $\neg a \wedge b$, $\neg a \wedge \neg b$ are pairwise disjoint and their supremum is 1,

$$M(a \wedge b) + M(a \wedge \neg b) + M(\neg a \wedge b) + M(\neg a \wedge \neg b) = 1,$$

and this allows us to rewrite Eq. (6) into Eq. (4). As all transformations are invertible, the two expressions are equivalent. □

Theorem 1. *A measure $M\colon 2^{2^U} \to \mathbb{R}_{\geq 0}$ is factorable if and only if*

$$M(u \wedge v) = M(u)M(v) \tag{7}$$

for all distinct $u, v \in U \cup \{\neg w \mid w \in U\}$ such that $u \neq \neg v$.

Proof. ($\Leftarrow$) For each $x \in U$, let $W_x\colon 2^{\{x\}} \to \mathbb{R}_{\geq 0}$ be defined by $W_x(\{x\}) = M(x)$ and $W_x(\emptyset) = M(\neg x)$. Let M_W be the measure induced by $W = \prod_{x \in U} W_x$. We will show that $M = M_W$. First, note that $M_w(\bot) = 0 = M(\bot)$ by Definition 3 and Lemma 1. Second, let $a = \bigwedge_{u \in U} a_u$ be an atom in 2^{2^U} such that $a_u \in \{u, \neg u\}$ for all $u \in U$. Then

$$M_W(a) = W(\{a_u \mid u \in U\}) = \prod_{u \in U} W_u(\{a_u\}) = \prod_{u \in U} M(a_u) = M\left(\bigwedge_{u \in U} a_u\right) = M(a)$$

Finally, note that if M_W and M agree on all atoms, then they must also agree on all other non-zero elements of the Boolean algebra.

($\Rightarrow$) For the other direction, we are given a factored weight function $W = \prod_{x \in U} W_x$, and we want to show that its induced measure M_W satisfies Eq. (7). Let $k_u, k_v \in U \cup \{\neg w \mid w \in U\}$ be such that $k_u \in \{u, \neg u\}$, $k_v \in \{v, \neg v\}$, and $u \neq v$. We then want to show that

$$M_W(k_u \wedge k_v) = M_W(k_u)M_W(k_v) \tag{8}$$

which is equivalent to

$$M_W(k_u \wedge k_v) \cdot M_W(\neg k_u \wedge \neg k_v) = M_W(k_u \wedge \neg k_v) \cdot M_w(\neg k_u \wedge k_v) \tag{9}$$

by Lemma 2. Then

$$\begin{aligned} M_W(k_u \wedge k_v) &= \sum_{\{a\} \leq k_u \wedge k_v} W(a) = \sum_{\{a\} \leq k_u \wedge k_v} \prod_{x \in U} W_x(a) \\ &= \sum_{\{a\} \leq k_u \wedge k_v} W_u(a_u) W_v(a_v) \prod_{x \in U \setminus \{u,v\}} W_x(a) = \sum_{\{a\} \leq k_u \wedge k_v} W_u(k_u) W_v(k_v) \prod_{x \in U \setminus \{u,v\}} W_x(a) \\ &= W_u(k_u) W_v(k_v) \sum_{\{a\} \leq k_u \wedge k_v} \prod_{x \in U \setminus \{u,v\}} W_x(a) = W_u(k_u) W_v(k_v) C, \end{aligned}$$

where $C = \sum_{\{a\} \leq k_u \wedge k_v} \prod_{x \in U \setminus \{u,v\}} W_x(a)$, and is the same for $M_W(\neg k_u \wedge k_v)$, $M_W(k_u \wedge \neg k_v)$, and $M_W(\neg k_u \wedge \neg k_v)$ as well. But then Eq. (9) becomes

$$W_u(k_u) W_v(k_v) W_u(\neg k_u) W_v(\neg k_v) C^2 = W_u(k_u) W_v(\neg k_v) W_u(\neg k_u) W_v(k_v) C^2$$

which is trivially true. □

4 Previous Work

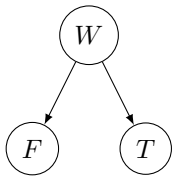
4.1 Using WMC to Perform Inference on Bayesian Networks

Hitherto, four techniques have been proposed for encoding Bayesian networks into instances of WMC. We will identify them based on the initials of authors as well as publications years: `d02` [9], `sbk05` [28], `cd05` [3], and `cd06` [4]. Below we friendly summarise the observed performance differences among them. Sang et al. [28] claim that `sbk05` is a smaller encoding than `d02` with respect to both the number of clauses and the number of variables but provide no experimental comparison. Chavira and Darwiche [3] compare `cd05` with `d02` by measuring the time it takes to compile either encoding into an arithmetic circuit (but do not measure inference time). The more recent encoding `cd05` always compiles faster and results in a smaller arithmetic circuit (as measured by the number of edges). In their subsequent paper, the same authors perform two sets of experiments (that are relevant to this summary) [4]. First, they compile `cd05` and `cd06` encodings into d-DNNF (i.e., deterministic decomposable negation normal form [8]), measuring both compilation time and numbers of edges in the d-DNNF diagram. The results are mostly in favour of `cd06`. Second, they compare the inference time of `sbk05` run with Cachet [27] with the compile times of `cd05` and `cd06`, but only on five (types of) instances. In these experiments, `cd06` is always faster than `cd05`, while the comparison with `sbk05` is mixed. Sometimes `cd06` is orders of magnitude faster than `sbk05`, sometimes slightly slower. The performance difference between `sbk05` and `cd05` is even harder to judge: `sbk05` is better on three out of five instances and worse on the remaining two. Based on this description, one would expect `cd06` to be faster than both `cd05` and `sbk05`, both of which should be faster than `d02`. The experiments in Section 6, however, strongly disagree with this prediction, showing that the quality of an encoding depends strongly on the underlying search algorithm or compilation technique.

4.2 Algebraic Decision Diagrams and Their Use in Probabilistic Inference

Cover:

- ADDs provide an efficient way to manipulate functions from BAs/power sets [1].
- background reading
  - Compiling Bayesian Networks Using Variable Elimination (Chavira and Darwiche) [5]



w	$\Pr(W = w)$
1	0.5
0	0.5

w	f	$\Pr(F = f \mid W = w)$
1	1	0.6
1	0	0.4
0	1	0.1
0	0	0.9

w	t	$\Pr(T = t \mid W = w)$
1	l	0.2
1	m	0.4
1	h	0.4
0	l	0.6
0	m	0.3
0	h	0.1

Figure 1: An example Bayesian network with its CPTs

– On the Relationship between Sum-Product Networks and Bayesian Networks [31]
– Using ROBDDs for Inference in Bayesian Networks with Troubleshooting as an Example [23]
– SPUDD: Stochastic Planning using Decision Diagrams [16]

5 Encoding Bayesian Networks Using Conditional Weights

In line with the names of previous encodings, we shall call this encoding `db20`.

A *Bayesian network* is a directed acyclic graph with random variables as vertices that defines a probability distribution over them. Let $\mathcal{V}$ denote the set of random variables and $X \in \mathcal{V}$ be a random variable. Let $\operatorname{im} X$ denote the set of possible values of X and $\operatorname{pa}(X)$ denote the set of its parents. The full probability distribution is then equal to

$$\prod_{X \in \mathcal{V}} \Pr(X \mid \operatorname{pa}(X)). \tag{10}$$

If all random variables are discrete (and we only consider discrete Bayesian networks in this paper), the probabilities contained in each factor of Eq. (10) can be summarised in a table known as a *conditional probability table* (CPT). See Fig. 1 for an example Bayesian network which we will refer to throughout this section. For this network, $\mathcal{V} = \{W, F, T\}$, $\operatorname{pa}(W) = \emptyset$, $\operatorname{pa}(F) = \operatorname{pa}(T) = \{W\}$, $\operatorname{im} W = \operatorname{im} F = \{0, 1\}$, and $\operatorname{im} T = \{l, m, h\}$.

Definition 4 (Indicator variables)**.** Let $X \in \mathcal{V}$ be a random variable. If X is binary (i.e., $|\operatorname{im} X| = 2$), we can arbitrary identify one of the values as 1 and the other one as 0 (i.e, $\operatorname{im} X \cong \{0, 1\}$). Then X can be represented by a single *indicator variable* $\lambda_{X=1}$. For notational simplicity, for any set S, we write $\lambda_{X=0} \in S$ or $S = \{\lambda_{X=0}, \dots\}$ to mean $\lambda_{X=1} \notin S$.

On the other hand, if X is not binary, we represent X with $|\operatorname{im} X|$ indicator variables, one for each value. We let

$$E(X) = \begin{cases} \{\lambda_{X=1}\} & \text{if } |\operatorname{im} X| = 2 \\ \{\lambda_{X=x} \mid x \in \operatorname{im} X\} & \text{otherwise.} \end{cases}$$

denote the set of indicator variables for X and $E^*(X) = E(X) \cup \bigcup_{Y \in \operatorname{pa}(X)} E(Y)$ denote the set of indicator variables for X and its parents in the Bayesian network. Finally, let $U = \bigcup_{X \in \mathcal{V}} E(X)$ denote the set of all indicator variables for all random variables in the Bayesian network.

For the Bayesian network in Fig. 1, this gives us:

$$\begin{aligned} E(W) &= \{\lambda_{W=1}\}, \\ E(F) &= \{\lambda_{F=1}\}, \\ E(T) &= \{\lambda_{T=l}, \lambda_{T=m}, \lambda_{T=h}\}, \\ E^*(W) &= \{\lambda_{W=1}\}, \\ E^*(F) &= \{\lambda_{F=1}, \lambda_{W=1}\}, \\ E^*(T) &= \{\lambda_{T=l}, \lambda_{T=m}, \lambda_{T=h}, \lambda_{W=1}\}. \end{aligned}$$

Algorithm 1: Encoding a Bayesian network as a function $2^U \to \mathbb{R}_{\geq 0}$

Data: a Bayesian network with vertices $\mathcal{V}$ and probability distribution Pr
Result: a function $\phi\colon 2^U \to \mathbb{R}_{\geq 0}$
$\phi \leftarrow 1$;
for $X \in \mathcal{V}$ **do**
  let $\mathrm{pa}(X) = \{Y_1, \ldots, Y_n\}$;
  $\mathrm{CPT}_X \leftarrow 0$;
  if $|\operatorname{im} X| = 2$ **then**
    for $(y_1, \ldots, y_n) \in \prod_{i=1}^n \operatorname{im} Y_i$ **do**
      $p_1 \leftarrow \Pr(X = 1 \mid Y_1 = y_1, \ldots, Y_n = y_n)$;
      $p_0 \leftarrow \Pr(X \neq 1 \mid Y_1 = y_1, \ldots, Y_n = y_n)$;
      $\mathrm{CPT}_X \leftarrow \mathrm{CPT}_X + p_1[\lambda_{X=1}] \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}] + p_0\overline{[\lambda_{X=1}]} \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}]$;
  else
    let $\operatorname{im} X = \{x_1, \ldots, x_m\}$;
    for $x \in \operatorname{im} X$ **and** $(y_1, \ldots, y_n) \in \prod_{i=1}^n \operatorname{im} Y_i$ **do**
      $p_x \leftarrow \Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n)$;
      $\mathrm{CPT}_X \leftarrow \mathrm{CPT}_X + p_x[\lambda_{X=x}] \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}] + \overline{[\lambda_{X=1}]} \cdot \prod_{i=1}^n [\lambda_{Y_i=y_i}]$;
    $\mathrm{CPT}_X \leftarrow \mathrm{CPT}_X \cdot (\sum_{i=1}^m [\lambda_{X=x_i}]) \cdot \prod_{i=1}^m \prod_{j=i+1}^m (\overline{[\lambda_{X=x_i}]} + \overline{[\lambda_{X=x_j}]})$;
  $\phi \leftarrow \phi \cdot \mathrm{CPT}_X$;
return ϕ;

Describe the algorithm and the CPT functions.

For the Bayesian network in Fig. 1, we have:

$$\begin{aligned} \mathrm{CPT_W} &= 0.5[\lambda_{W=1}] + 0.5\overline{[\lambda_{W=1}]} = 0.5 \cdot 1, \\ \mathrm{CPT_F} &= 0.6[\lambda_{F=1}] \cdot [\lambda_{W=1}] + 0.4[\lambda_{F=0}] \cdot [\lambda_{W=1}] + 0.1[\lambda_{F=1}] \cdot [\lambda_{W=0}] + 0.9[\lambda_{F=0}] \cdot [\lambda_{W=0}] \\ &= 0.6[\lambda_{F=1}] \cdot [\lambda_{W=1}] + 0.4\overline{[\lambda_{F=1}]} \cdot [\lambda_{W=1}] + 0.1[\lambda_{F=1}] \cdot \overline{[\lambda_{W=1}]} + 0.9\overline{[\lambda_{F=1}]} \cdot \overline{[\lambda_{W=1}]}, \\ \mathrm{CPT_T} &= ([\lambda_{T=l}] + [\lambda_{T=m}] + [\lambda_{T=h}]) \cdot (\overline{[\lambda_{T=l}]} + \overline{[\lambda_{T=m}]}) \cdot (\overline{[\lambda_{T=l}]} + \overline{[\lambda_{T=h}]}) \cdot (\overline{[\lambda_{T=m}]} + \overline{[\lambda_{T=h}]}) \cdot (\ldots). \end{aligned}$$

5.1 Proof of Correctness

Lemma 3. *Let $X \in \mathcal{V}$ be a random variable with parents $\mathrm{pa}(X) = \{Y_1, \ldots, Y_n\}$. Then $\mathrm{CPT}_X\colon 2^{E^*(X)} \to \mathbb{R}_{\geq 0}$ is such that for any $x \in \operatorname{im} X$ and $(y_1, \ldots, y_n) \in \prod_{i=1}^n \operatorname{im} Y_i$,*

$$\mathrm{CPT}_X(\{\lambda_{X=x}\} \cup \{\lambda_{Y_i=y_i} \mid i = 1, \ldots, n\}) = \Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n).$$

Proof. Let $\tau = \{\lambda_{X=x}\} \cup \{\lambda_{Y_i=y_i} \mid i = 1, \ldots, n\}$. If X is binary, then CPT_X is a sum of $2\prod_{i=1}^{n} |\operatorname{im} Y_i|$ terms, one for each possible assignment of values to variables $X, Y_1, \ldots, Y_n$. Exactly one of these terms is nonzero when applied to τ, and it is equal to $\Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n)$ by definition.

If X is not binary, then $\left(\sum_{i=1}^{m} [\lambda_{X=x_i}]\right)(\tau) = 1$, and

$$\left(\prod_{i=1}^{m}\prod_{j=i+1}^{m} (\overline{[\lambda_{X=x_i}]} + \overline{[\lambda_{X=x_j}]})\right)(\tau) = 1,$$

so $\mathrm{CPT}_X(\tau) = \Pr(X = x \mid Y_1 = y_1, \ldots, Y_n = y_n)$ by a similar argument as before. □

Proposition 1. *$\phi\colon 2^U \to \mathbb{R}_{\geq 0}$ represents the full probability distribution of the Bayesian network, i.e., if $\mathcal{V} = \{X_1, \ldots, X_n\}$, then*

$$\phi(\tau) = \begin{cases} \Pr(X_1 = x_1, \ldots, X_n = x_n) & \text{if } \tau = \{\lambda_{X_i=x_i} \mid i = 1, \ldots, n\} \text{ for some } (x_1, \ldots, x_n) \in \prod_{i=1}^{n} \operatorname{im} X_i \\ 0 & \text{otherwise,} \end{cases}$$

for all $\tau \in 2^U$.

Proof. If $\tau = \{\lambda_{X=v_X} \mid X \in \mathcal{V}\}$ for some $(v_X)_{X\in\mathcal{V}} \in \prod_{X\in\mathcal{V}} \operatorname{im} X$, then

$$\phi(\tau) = \prod_{X\in\mathcal{V}} \Pr\left(X = v_X \,\middle|\, \bigwedge_{Y\in\mathrm{pa}(X)} Y = v_Y\right) = \Pr\left(\bigwedge_{X\in\mathcal{V}} X = v_X\right)$$

by Lemma 3 and the definition of a Bayesian network. Otherwise there must be some non-binary random variable $X \in \mathcal{V}$ such that $|E(X) \cap \tau| \neq 1$. If $E(X) \cap \tau = \emptyset$, then $\left(\sum_{i=1}^{m} [\lambda_{X=x_i}]\right)(\tau) = 0$, and so $\mathrm{CPT}_X(\tau) = 0$, and $\phi(\tau) = 0$. If $|E(X) \cap \tau| > 1$, then we must have two different values $x_1, x_2 \in \operatorname{im} X$ such that $\{\lambda_{X=x_1}, \lambda_{X=x_2}\} \subseteq \tau$ which means that $(\overline{[\lambda_{X=x_1}]} + \overline{[\lambda_{X=x_2}]})(\tau) = 0$, and so, again, $\mathrm{CPT}_X(\tau) = 0$, and $\phi(\tau) = 0$. □

Theorem 2. *Let $\phi\colon 2^U \to \mathbb{R}_{\geq 0}$ be the function generated by Algorithm 1. Then $(\exists_U(\phi \cdot [\lambda_{X=x}]))(\emptyset) = \Pr(X = x)$.*

Proof. Let $\mathcal{V} = \{X, Y_1, \ldots, Y_n\}$. Then

$$\begin{aligned}(\exists_U(\phi \cdot [\lambda_{X=x}]))(\emptyset) &= \sum_{\tau\in 2^U} (\phi \cdot [\lambda_{X=x}])(\tau) = \sum_{\lambda_{X=x}\in\tau\in 2^U} \phi(\tau) = \sum_{\lambda_{X=x}\in\tau\in 2^U} \left(\prod_{Y\in\mathcal{V}} \mathrm{CPT}_Y\right)(\tau) \\ &= \sum_{(y_1,\ldots,y_n)\in\prod_{i=1}^{n}\operatorname{im} Y_i} \Pr(X = x, Y_1 = y_1, \ldots, Y_n = y_n) = \Pr(X = x)\end{aligned}$$

by the following arguments:

- the proof of Theorem 1 in the ADDMC paper [12];
- if $\lambda_{X=x} \notin \tau \in 2^U$, then $(\phi \cdot [\lambda_{X=x}])(\tau) = \phi(\tau) \cdot [\lambda_{X=x}](\tau \cap \{\lambda_{X=x}\}) = \phi(\tau) \cdot 0 = 0$;
- Proposition 1;
- marginalisation of a probability distribution.

□

5.2 Textual Representation

Describe how this is an intermediate step before we get the ϕ from the algorithm. Each CPT is treated as a clause.

The Bayesian network in Fig. 1 can be represented in a textual format as

$\lambda_{T=l}$	$\lambda_{T=m}$	$\lambda_{T=h}$		0
	$-\lambda_{T=l}$	$-\lambda_{T=m}$		0
	$-\lambda_{T=l}$	$-\lambda_{T=h}$		0
	$-\lambda_{T=m}$	$-\lambda_{T=h}$		0
w	$\lambda_{W=1}$		0.5	0.5
w	$\lambda_{F=1}$	$\lambda_{W=1}$	0.6	0.4
w	$\lambda_{F=1}$	$-\lambda_{W=1}$	0.1	0.9
w	$\lambda_{T=l}$	$\lambda_{W=1}$	0.2	1
w	$\lambda_{T=m}$	$\lambda_{W=1}$	0.4	1
w	$\lambda_{T=h}$	$\lambda_{W=1}$	0.4	1
w	$\lambda_{T=l}$	$\lambda_{W=0}$	0.6	1
w	$\lambda_{T=m}$	$\lambda_{W=0}$	0.3	1
w	$\lambda_{T=h}$	$\lambda_{W=0}$	0.1	1

with each λ replaced with a unique positive integer. This format is based on the format used by the Cachet solver [27] to encode WMC problems, which extends the DIMACS format for conjunctive normal form (CNF) formulas with weight clauses. Subsequently, we extend it in two ways:

- a single weight clause now supports an arbitrary number of literals,
- and each weight clause has two probabilities instead of one (i.e., we no longer assume that $\Pr(v) + \Pr(\neg v) = 1$ for all variables $v \in U$).

The way we use this encoding, it is always the case that either both probabilities sum to one, or the second probability (i.e., the probability for the complement of the variable) is equal to one.

5.3 Changes to ADDMC

ADDMC constructs the *Gaifman graph* [14] of the input CNF formula as an aid for the algorithm's heuristics. This graph has as vertices the variables of the formula, and there is an edge between two variables u and v if there is a clause in the formula that contains both u and v. We extend this definition to functions on Boolean algebras, i.e., the factors of ϕ. For any pair of distinct variables $u, v \in U$, we draw an edge between them in the Gaifman graph if there is a function $A\colon 2^X \to \mathbb{R}_{\geq 0}$ that is a factor of ϕ such that $u \in X$ and $v \in X$. For instance, a factor such as CPT_X will enable edges between all distinct pairs of variables in $E^*(X)$.

This needs rewriting.

ADDMC multiplies the final answer by $w(u) + w(-u)$ for every $u \in U$ that was 'simplified out' of the ADD for ϕ and so is not featured in the $\exists_U$ projection. In our format, for every such $u \in U$, we multiply the answer by two. Each such $u \in U$ must satisfy two properties:

- all the probabilities/weights associated with u are equal to 0.5 (otherwise the corresponding CPT would depend on u),
- and all other CPTs are independent of u (or they may have a trivial dependence, where the probability stays the same if u is replaced with its complement).

Thus, the CPT that corresponds to u still multiplies every model by 0.5, but the number of models being considered by the ADDMC is halved. To correct for this, we need to perform this multiplication.

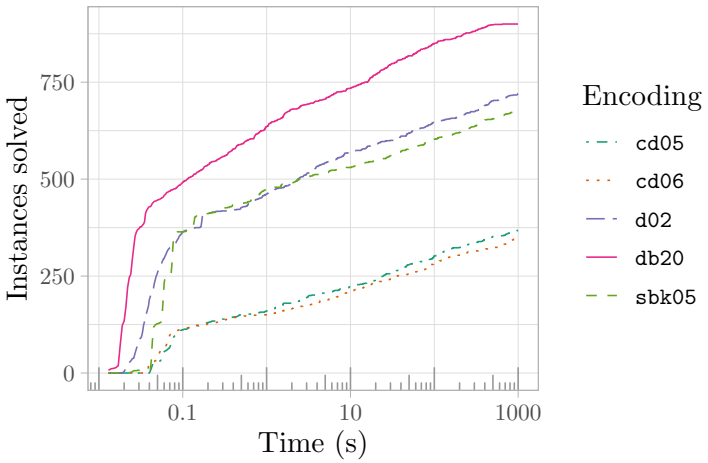


Figure 2: Cumulative number of instances solved by ADDMC over time using each encoding

Table 3: The numbers of instances solved by ADDMC with each encoding (uniquely, faster than with others, and in total) under the described time and memory constraints (out of 1216 instances)

Encoding	Unique	Fastest	Total
cd05	2	3	372
cd06	0	1	351
d02	41	99	726
db20	228	871	901
sbk05	6	36	687

6 Experimental Comparison

- We chose not to measure or compare encoding time because of different languages of implementation and the fact that our encoding algorithm (i.e., the process that converts a Bayesian network into a textual encoding such as in Section 5.2) is linear in the total number of CPT rows and so is unlikely to be slower than, e.g., cd06, which relies on solving NP-complete problems as part of the encoding process [4].
- Whenever a Bayesian network comes with an evidence file, we compute the probability of evidence. Otherwise, let X denote the last-mentioned vertex in the Bayesian network. If true is a valid value of X, we compute the marginal probability of $X =$ true. Otherwise, we pick the value of X which is listed first and calculate its marginal probability.
- The experiments were run on Intel Xeon Gold 6138 processor with an 8 GB memory limit.
- For all other encodings, we use their implementation in Ace 3.0[2] with -encodeOnly and -noEclause flags. However, Ace was not used to encode evidence, as preliminary experiments revealed that the evidence-encoding implementation contains bugs that can lead to incorrect answers or a Java exception being thrown on some instances of the data set (and the source code is not publicly available). Instead, we simply list all the evidence as additional clauses in the encoding (regardless of which encoding is used).
- Note that cd05 and cd06 purposefully produce overly relaxed encodings that contain extra models and thus yield incorrect probabilities [3, 4]. These additional models are supposed to be filtered out during circuit compilation [3], but this is not easily achievable with ADDMC. Nonetheless, we include both encodings in our timing experiments.

Data. For experiments, we use the Bayesian networks available with Ace and Cachet[3], most of which happen to be binary. We classify them into the following seven categories: • DQMR and • Grid networks as described by Sang et al. [28], • Friends and Smokers, • Mastermind, and • Random Blocks from the work of Chavira et al. [7], • remaining binary Bayesian networks that include Plan Recognition [28], Students

[2] http://reasoning.cs.ucla.edu/ace/
[3] https://www.cs.rochester.edu/u/kautz/Cachet/

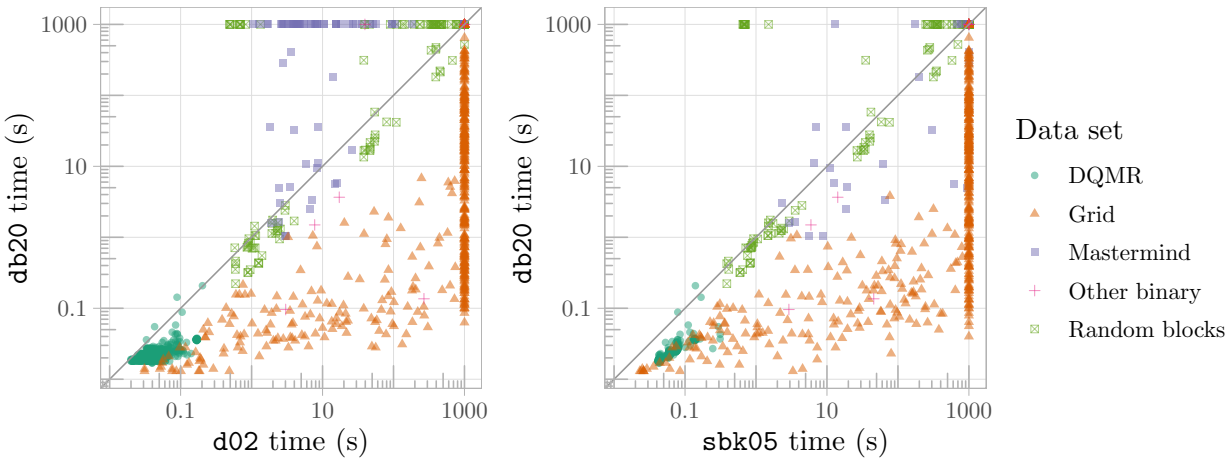


Figure 3: ADDMC inference time using `db20` compared to `d02` (left) and `sbk05` (right) on an instance-by-instance basis across all data sets.

Table 4: Asymptotic upper bounds on the numbers of variables and clauses/ADDs for each encoding

Encoding(s)	Variables	Clauses/ADDs
`cd05`, `cd06`, `sbk05`	$\mathcal{O}(nv^{d+1})$	$\mathcal{O}(nv^{d+1})$
`d02`	$\mathcal{O}(nv^{d+1})$	$\mathcal{O}(ndv^{d+1})$
`db20`	$\mathcal{O}(nv)$	$\mathcal{O}(nv^2)$

and Professors [7], and `tcc4f`, and • non-binary classic Bayesian networks (`alarm`, `diabetes`, `hailfinder`, `mildew`, `munin1`–4, `pathfinder`, `pigs`, `water`).

Observations.

- The order of the encodings from best to worst is exactly the opposite of that in the previous literature.
- Our encoding is particularly promising on instances of Grid networks. They are set up so that the entire Bayesian network is relevant to the query.
- However, `db20` struggles with instances from Mastermind and Random Blocks domains (but so does `sbk05`). We conjecture that this is so either because the particular structure of these problems confuse the heuristics used by ADDMC or because these instances allow for significant simplifications that are exploited by `d02` but not `db20`.
- Everything `d02` can do in 1000 s, `db20` can do in 10 s (find the exact number).

Explaining The Performance Benefits. Let $n = |\mathcal{V}|$ be the number of vertices in the Bayesian network, $d = \max_{X \in \mathcal{V}} |\mathrm{pa}(X)|$ the maximum in-degree (i.e., number of parents), and $v = \max_{X \in \mathcal{V}} |\operatorname{im} X|$ the maximum number of values per variable. Then... Table 4.

Remember that these are upper bounds! Some of these encoding have a lot of optimisations that are not captured by three parameters!

Explain why clauses and ADDs are alike.

7 Conclusion and Future Work

- Bayesian networks and ADDMC are only particular examples. This should also work with Cachet.
- Extra benefit: one does not need to come up with a way to turn some probability distribution to into a fully independent one.
- Important future work: replacing ADDs with AADDs [29] is likely to bring performance benefits. Other extensions:
  - FOADDs can represent first order statements;
  - XADDs can replace WMI for continuous variables;
  - ADDs with intervals can do approximations.
- Filtering out ADDs that have nothing to do with the answer helps tremendously, but I'm purposefully not doing that. Perhaps a heuristic could do the same thing?
- Encodings for everything else
  - probabilistic programs [17]
  - ProbLog [13]
    - proof-based [22]
    - rule-based [18]
- Bayesian networks are often solved in a compile once, query many times fashion. This can be achieved using ADDMC by selecting a subset S of variable we may want to query over and running ADDMC while excluding S from variable elimination/projection/$\exists$.
- I could also add a memory usage cumulative plot, but it doesn't add any extra info except for the fact that inference is memory-intensive.

Acknowledgements. This work has made use of the resources provided by the Edinburgh Compute and Data Facility (ECDF) (`http://www.ecdf.ed.ac.uk/`).

References

[1] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. *Formal Methods Syst. Des.*, 10(2/3):171–206, 1997.

[2] Elena Castiñeira, Susana Cubillo, and Enric Trillas. On possibility and probability measures in finite Boolean algebras. *Soft Comput.*, 7(2):89–96, 2002.

[3] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks with local structure. In Kaelbling and Saffiotti [19], pages 1306–1312.

[4] Mark Chavira and Adnan Darwiche. Encoding CNFs to empower component analysis. In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 2006.

[5] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks using variable elimination. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2443–2449, 2007.

[6] Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artif. Intell.*, 172(6-7):772–799, 2008.

[7] Mark Chavira, Adnan Darwiche, and Manfred Jaeger. Compiling relational Bayesian networks for exact inference. *Int. J. Approx. Reason.*, 42(1-2):4–20, 2006.

[8] Adnan Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *J. Appl. Non Class. Logics*, 11(1-2):11–34, 2001.

[9] Adnan Darwiche. A logical approach to factoring belief networks. In Dieter Fensel, Fausto Giunchiglia, Deborah L. McGuinness, and Mary-Anne Williams, editors, *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002*, pages 409–420. Morgan Kaufmann, 2002.

[10] Adnan Darwiche. New advances in compiling CNF into decomposable negation normal form. In de Mántaras and Saitta [11], pages 328–332.

[11] Ramón López de Mántaras and Lorenza Saitta, editors. *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*. IOS Press, 2004.

[12] Jeffrey M. Dudek, Vu Phan, and Moshe Y. Vardi. ADDMC: weighted model counting with algebraic decision diagrams. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1468–1476. AAAI Press, 2020.

[13] Daan Fierens, Guy Van den Broeck, Ingo Thon, Bernd Gutmann, and Luc De Raedt. Inference in probabilistic logic programs using weighted CNF's. In Fábio Gagliardi Cozman and Avi Pfeffer, editors, *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, pages 211–220. AUAI Press, 2011.

[14] Haim Gaifman. On local and non-local properties. In *Studies in Logic and the Foundations of Mathematics*, volume 107, pages 105–135. Elsevier, 1982.

[15] Theodore Hailperin. Probability logic. *Notre Dame Journal of Formal Logic*, 25(3):198–212, 1984.

[16] Jesse Hoey, Robert St-Aubin, Alan J. Hu, and Craig Boutilier. SPUDD: stochastic planning using decision diagrams. In Kathryn B. Laskey and Henri Prade, editors, *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 279–288. Morgan Kaufmann, 1999.

[17] Steven Holtzen, Guy Van den Broeck, and Todd D. Millstein. Dice: Compiling discrete probabilistic programs for scalable inference. *CoRR*, abs/2005.09089, 2020.

[18] Tomi Janhunen. Representing normal programs with clauses. In de Mántaras and Saitta [11], pages 358–362.

[19] Leslie Pack Kaelbling and Alessandro Saffiotti, editors. *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*. Professional Book Center, 2005.

[20] Peter H. Krauss. Representation of conditional probability measures on Boolean algebras. *Acta Mathematica Hungarica*, 19(3-4):229–241, 1968.

[21] Jean-Marie Lagniez and Pierre Marquis. An improved decision-DNNF compiler. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 667–673. ijcai.org, 2017.

[22] Theofrastos Mantadelis and Gerda Janssens. Dedicated tabling for a probabilistic setting. In Manuel V. Hermenegildo and Torsten Schaub, editors, *Technical Communications of the 26th International Conference on Logic Programming, ICLP 2010, July 16-19, 2010, Edinburgh, Scotland, UK*, volume 7 of *LIPIcs*, pages 124–133. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

[23] Thomas D. Nielsen, Pierre-Henri Wuillemin, Finn Verner Jensen, and Uffe Kjærulff. Using ROBDDs for inference in bayesian networks with troubleshooting as an example. In Craig Boutilier and Moisés Goldszmidt, editors, *UAI '00: Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence, Stanford University, Stanford, California, USA, June 30 - July 3, 2000*, pages 426–435. Morgan Kaufmann, 2000.

[24] Nils J. Nilsson. Probabilistic logic. *Artif. Intell.*, 28(1):71–87, 1986.

[25] Nils J. Nilsson. Probabilistic logic revisited. *Artif. Intell.*, 59(1-2):39–42, 1993.

[26] Umut Oztok and Adnan Darwiche. A top-down compiler for sentential decision diagrams. In Yang and Wooldridge [30], pages 3141–3148.

[27] Tian Sang, Fahiem Bacchus, Paul Beame, Henry A. Kautz, and Toniann Pitassi. Combining component caching and clause learning for effective model counting. In *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings*, 2004.

[28] Tian Sang, Paul Beame, and Henry A. Kautz. Performing Bayesian inference by weighted model counting. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 475–482. AAAI Press / The MIT Press, 2005.

[29] Scott Sanner and David A. McAllester. Affine algebraic decision diagrams (aadds) and their application to structured probabilistic inference. In Kaelbling and Saffiotti [19], pages 1384–1390.

[30] Qiang Yang and Michael J. Wooldridge, editors. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. AAAI Press, 2015.

[31] Han Zhao, Mazen Melibari, and Pascal Poupart. On the relationship between sum-product networks and Bayesian networks. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 116–124. JMLR.org, 2015.